# SlotSwapper – Step-by-Step Implementation Guide

## 1. Project Setup

- Choose your tech stack: React/Vue/Angular for frontend, Node/Express or Django/FastAPI for backend.

- Initialize frontend and backend projects.

- Set up GitHub repository and create separate folders for frontend and backend.

- Install dependencies for both sides.

## 2. User Authentication (Backend)

- Create User model with fields: name, email, password.

- Implement Sign Up API.

- Hash passwords using bcrypt.

- Implement Login API.

- Generate JWT token for authenticated requests.

- Create JWT middleware for protected routes.

## 3. Database & Calendar Data Model

- Create Event model with: title, startTime, endTime, status, userId.

- Define event statuses: BUSY, SWAPPABLE, SWAP\_PENDING.

- Implement CRUD APIs for events (Create, Read, Update, Delete).

## 4. Backend Swap Logic

- Implement GET /api/swappable-slots to fetch swappable events from other users.

- Create SwapRequest model with fields: mySlotId, theirSlotId, requesterId, ownerId, status.

- Implement POST /api/swap-request API to create swap requests.

- Validate slot availability and update event statuses to SWAP\_PENDING.

- Implement POST /api/swap-response/:requestId API.

- If rejected: reset event statuses to SWAPPABLE and update request status.

- If accepted: swap slot owners, set events to BUSY, update request status.

## 5. Frontend Implementation

- Build Sign Up and Login pages.

- Store JWT token in local storage.

- Create Calendar/Dashboard UI to show user events.

- Implement event creation and status update UI.

- Create Marketplace page to view swappable slots.

- Add ability to request swap by selecting user’s own swappable slot.

- Create Notifications page for Incoming and Outgoing swap requests.

- Implement Accept/Reject buttons with live updates.

## 6. State Management

- Use Redux/Context API/Pinia/Vuex for managing global state.

- Protect authenticated routes.

- Update calendar and request lists dynamically after every action.

## 7. Bonus Features (Optional)

- Add unit/integration tests for APIs.

- Implement real-time notifications using WebSockets.

- Deploy backend (Render/Heroku) and frontend (Vercel/Netlify).

- Add Dockerfile and docker-compose for easy setup.

## 8. Deliverables Checklist

- Public GitHub repository link.

- README.md including setup instructions, API documentation, challenges faced.

- Optional live demo link.